CSE 3105 / CSE 3137

OBJECT-ORIENTED ANALYSIS AND DESIGN

FALL 2024

COURSE PROJECT: *LIBRARY MANAGEMENT SYSTEM*

System Design Document

Group 19

Sıla Naz ASLAN 210316072

Sena KILINÇ 210316036

Aslı ÇELİK 210316024

Zafer SARIK 200316034

İsmail Ertuğrul BOZKURT 210316059

Umut Akbaş 210316010

28 December 2024

Table of Contents

1 Introduction

The library study desk reservation system helps students and visitors easily find and book study desks. This document provides an overview of the system’s design and goals, including making it user-friendly, offering real-time desk availability updates, ensuring data security, and being both scalable and cost-efficient. It also highlights the system’s structure, the challenges from existing systems, and how it enhances desk management efficiency to better meet user demands.

This document explains how the system assists in managing desk usage, avoiding booking conflicts, and improving resource fairness. Key features, such as "Checking Desk Availability" and "Making Reservations," are outlined in Section 3.1. These features help reduce errors and create a smoother experience for both users and administrators.

The design of the system makes it simple for users to book desks while ensuring equal access to resources during peak times, such as exam periods. It also includes tools for library staff to monitor desk utilization and optimize resource allocation, ensuring the library operates more effectively.

1.1 Design goals

High *Priority* - Performence

Library system handles peak usage times, such as during exams, when many users might access the system simultaneously.

High *Priority* - Ease of Use

Library system interface should be easy to use and understandable so that students do not experience any confusion or loss of time when making a reservation.

***Medium Priority* - Data security**

Library system handle sensitive data lie ID no, and maintaining trust requires robust security measures.

***Low Priority* - Cost efficiency**

Libraries often operate on tight budgets also this system is developed by students. Develop within budget constraints without compromising on critical features and user satisfaction.

***Low Priority* – Scalability**

Library system should be scalable, ready for future growth while concentrating on present needs

Rationale

Features vs. Complexity: Adding more features, such as waitlists or notifications, increases functionality but risks complicating the system. Keeping the interface intuitive may require limiting initial features to maintain usability and simplicity.

Security vs. Usability: Robust security measures are essential to protect user data and system integrity, but they can make the system less convenient for users. A balance is needed to ensure security does not interfere with ease of access.

Cost vs. Quality: High-quality systems provide reliability and rich features, but budget constraints might require prioritizing critical functionalities over advanced or premium options to stay within financial limits.
Scalability vs. Performance: Prioritizing scalability ensures future growth but may initially affect response times or complexity, balancing current performance with long-term needs.

2 Current Software Architecture

2. Current Software Architecture

The current library system is a basic setup that allows students to reserve individual rooms. However, it lacks a mechanism to track the occupancy of shared tables or provide a reservation function for these areas. This deficiency leads to significant time loss for students, especially during exam periods. The following aspects have been evaluated to analyze the current state of the system in detail:

2.1 General Overview of the System

- **Functionality:** The current system allows students to reserve individual rooms but does not offer any control or reservation mechanism for shared tables.
- **Limitations:**
    - The occupancy status of shared tables is not visible.
    - It is not possible to reserve tables in advance.
    - Students waste time searching for available tables.

2.2 Current Software Components

- **User Interface (Frontend):** Students use an interface to reserve individual rooms. However, this interface does not support tracking or reserving shared tables.
- **Database:**
    - Stores user information and individual room reservations.
    - Contains no data regarding shared tables.
- **Connection Logic (Backend):**
    - Provides basic functionality for user login and processing individual reservation requests.

2.3 Data Flow

The data flow in the current system is as follows:

1. **User Login:** The student logs into the system by entering their credentials.

2. **Reservation Request:** The user makes a reservation request for an individual room.
3. **Data Storage:** The reservation information is stored in the database.
4. **Physical Search:** Checking shared tables and finding an available table is entirely dependent on the student physically walking around the library.

2.4 Challenges and Limitations

The primary issues encountered in the current system are as follows:

- **Time Loss:** Physically checking shared tables to find availability is time-consuming.
- **Lack of Accessibility:** Users cannot remotely check or reserve tables.
- **Efficiency Problems:** Inefficient use of study areas reduces the overall productivity of the library.

2.5 Improvements in the Proposed System

The proposed system addresses these limitations by introducing the following features:

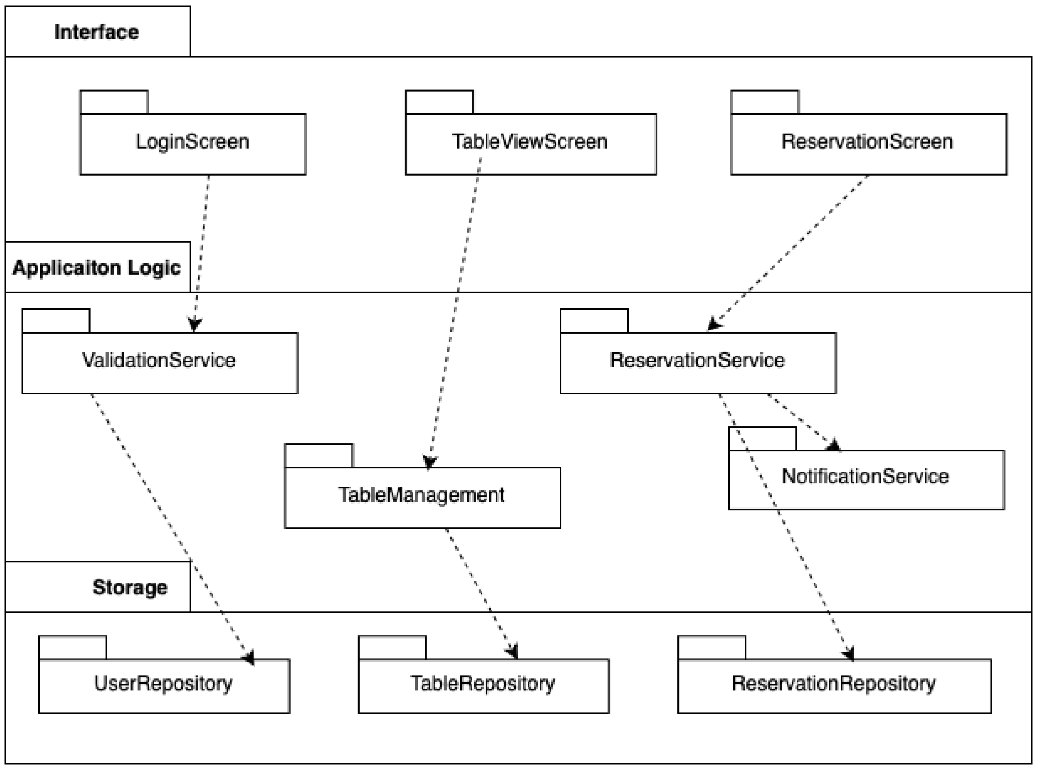
- **Real-time Occupancy Tracking: Users can view the current availability of shared tables in real time.**
- **Table Reservation Mechanism: Students can reserve shared tables in advance, similar to individual room reservations.**
- **Automated Notifications: Notifications alert users about their reservation start and end times, improving time management.**
- **Streamlined Interface: A user-friendly interface for managing shared tables.**
- **Efficient Data Management: The database now includes shared table information, enabling better tracking and analytics.**

2.6 Overall Assessment

The current system provides basic functionality for reserving individual rooms but lacks any control or monitoring mechanism for shared tables. These deficiencies reduce student satisfaction and hinder the efficient use of library resources. The proposed system overcomes these challenges by implementing advanced features such as real-time tracking, streamlined reservations, and automated notifications, greatly enhancing the library experience and resource efficiency.

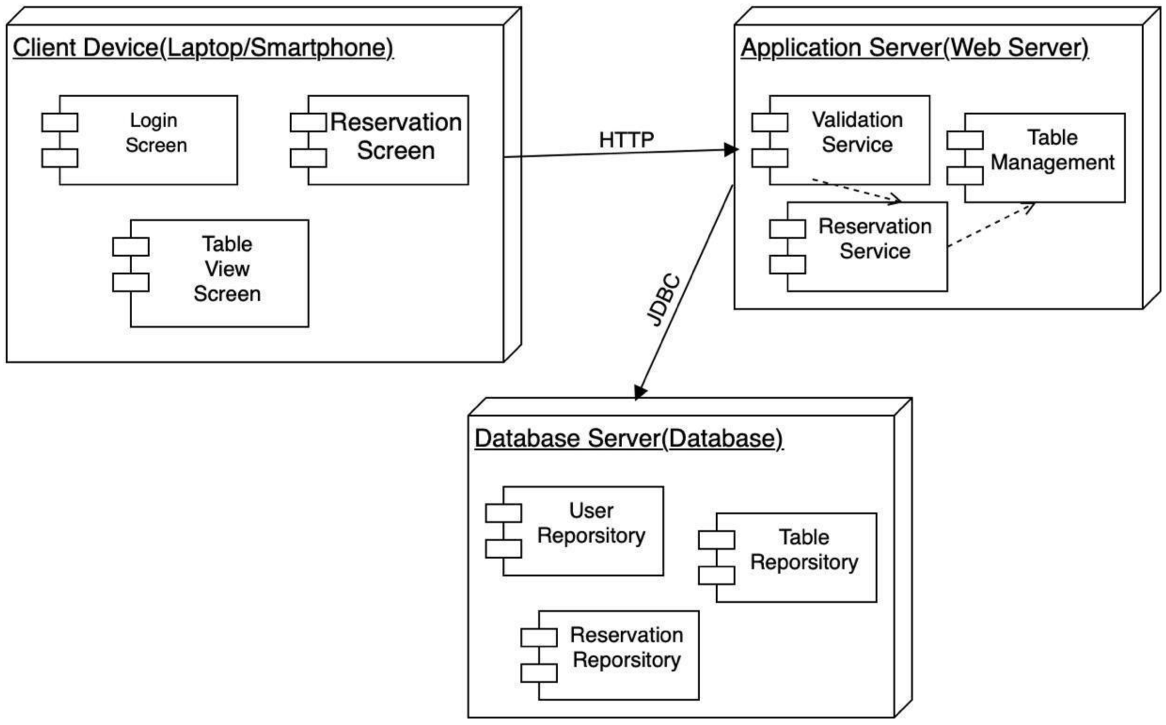
3 Proposed Software Architecture

3.1 Subsystem decomposition



This design follows the Layered Architecture Pattern, dividing the system into three layers: Interface, Application Logic, and Storage. The Interface Layer includes screens like LoginScreen, TableViewScreen, and ReservationScreen for user interaction. The Application Logic Layer contains services such as ValidationService for input verification, TableManagement for handling table status, and ReservationService for managing reservations, supported by NotificationService for sending alerts. Finally, the Storage Layer comprises repositories like UserRepository, TableRepository, and ReservationRepository to manage data. This architecture ensures modularity, maintainability, and clear separation of concerns, making the system scalable and easy to test.

3.2 Hardware/software mapping



3.3 Persistent data management

This section outlines how the system handles persistent data, including data storage strategies, database selection, and encapsulation. It ensures that the data required by the system is stored reliably and is accessible whenever needed.

1. Description of Persistent Data

The system will store the following types of persistent data:

- User Data:

Purpose: To authenticate users and manage their profiles.

Examples: Username, password (stored securely using encryption), email address, and user role (e.g., admin, student).

- Reservation Data:

Purpose: To track table reservations made by users and manage availability in real time.

Examples: Reservation ID, table number, reservation start and end times, user ID (foreign key).

- Table Information:

Purpose: To display the availability of tables in the library.

Examples: Table ID, status (e.g., available, reserved), and associated reservation details.

- System Logs:

Purpose: To maintain a record of user activity for debugging and audit purposes.

Examples: Login attempts, reservation changes, error reports.

This data is critical for system operations and must persist across sessions to maintain functionality and user experience.

2. Data Schemas

The database is structured using an Entity-Relationship (ER) Diagram to define the relationships between data entities. Below are the main entities and their relationships:

- Entities:

User: Contains user-specific data (UserID, Username, Email, Password).

Reservation: Tracks reservation details (ReservationID, UserID, TableID, StartTime, EndTime).

Table: Represents library tables (TableID, Status).

- Relationships:

A User can make multiple Reservations.

Each Reservation is linked to a specific Table.

3. Database Selection

The project uses a Relational Database Management System (RDBMS) like MySQL or PostgreSQL. These are chosen for the following reasons:

-Structured Data: The system requires well-defined relationships between entities.

-Consistency: ACID compliance ensures data integrity during transactions (e.g., making reservations).

-Scalability: Supports scaling for larger libraries or user bases in the future.

-Community Support: Widely adopted with extensive documentation and community assistance.

4. Data Management Infrastructure

To access and manage the database, the system employs a Data Access Layer (DAL) with the following features:

Encapsulation of Database Queries: All interactions with the database occur through an abstraction layer. This ensures modularity and reduces dependency on specific database implementations.

Access Control: Role-based access ensures that only authorized users can modify sensitive data. Regular users can only view or create their reservations. Admin users can modify table status or manage user accounts.

5. Database Management Requirements

The system incorporates several measures to ensure the reliability and security of stored data: Backup and Recovery: Regular database backups are scheduled to prevent data loss in case of system failures, recovery mechanisms ensure minimal downtime during restoration.

Data Encryption: Sensitive data, such as passwords, is hashed using algorithms like bcrypt or SHA-256. Data in transit is encrypted using SSL/TLS protocols.

Access Rights: Only the application server can directly interact with the database. Database credentials are stored securely, ensuring unauthorized access is prevented.

6. Examples of Artifacts

To support the above explanations, the following diagrams and schemas should be included:

-**Entity-Relationship Diagram:**

A visual representation of data entities (User, Reservation, Table) and their relationships.

-Database Schema:

-Users Table: UserID (Primary Key), Username, Password (Encrypted), Email

-Reservations Table: ReservationID (Primary Key), UserID (Foreign Key), TableID (Foreign Key), StartTime, EndTime.

-Tables Table: TableID (Primary Key), Status (Available, Reserved).

Conclusion

This section ensures the system's data is reliably stored, managed, and accessed. The selected RDBMS and DAL framework provide consistency, scalability, and security, making them suitable for a library management system. The encapsulated data handling ensures modularity and adaptability for future enhancements.

3.4 Access Control and Security

The **Access Control and Security** section describes how the Library Management System (LMS) manages user roles, permissions, and security protocols to ensure data integrity, confidentiality, and availability.

Access Matrix

The Access Matrix defines the relationship between **Student Actions, System Actions** and **System Resources (Reservations, Table Management, Notifications)**.

Resource	Login System	Table Status	Reservations	Notifications	System Settings
Student Actions	Login(), Logout()	ViewAvailableTables()	ReserveTable(), UpdateReservation()	ReceiveNotifications()	-
System Actions	ManageUsers(), ResetPassword()	UpdateTableStatus()	CancelReservation(), OverrideReservation()	SendNotifications()	ConfigureSettings()

Authentication Mechanisms

- **Username & Password Authentication:** Ensures only authorized users can access the system.
- **Session Management:** Tracks active user sessions to prevent unauthorized access.
- **Password Policies:** Enforces strong password rules for added security.

Authorization Mechanisms

- **Role-Based Access Control (RBAC):** Permissions are granted based on predefined user roles (Student, Administrator).
- **Access Control Lists (ACL):** Define specific actions each role can perform on system resources.

Encryption Protocols

- **SSL/TLS Encryption:** All data transmitted between clients and the server is encrypted.
- **Database Encryption:** Sensitive user information is stored using hashing and encryption algorithms.

Key Management

- Secure storage and regular rotation of encryption keys.
- Access to keys restricted to system administrators.

Security Policies

- Failed login attempt lockout.
- Regular security audits.
- Real-time alerts for suspicious activities.
-

3.5 Boundary Conditions

Boundary conditions describe how the system behaves during startup, shutdown, and error scenarios.

Start-Up Process:

- Initialize system components.
- Validate system dependencies and ensure all services are running.
- Log system status and readiness.

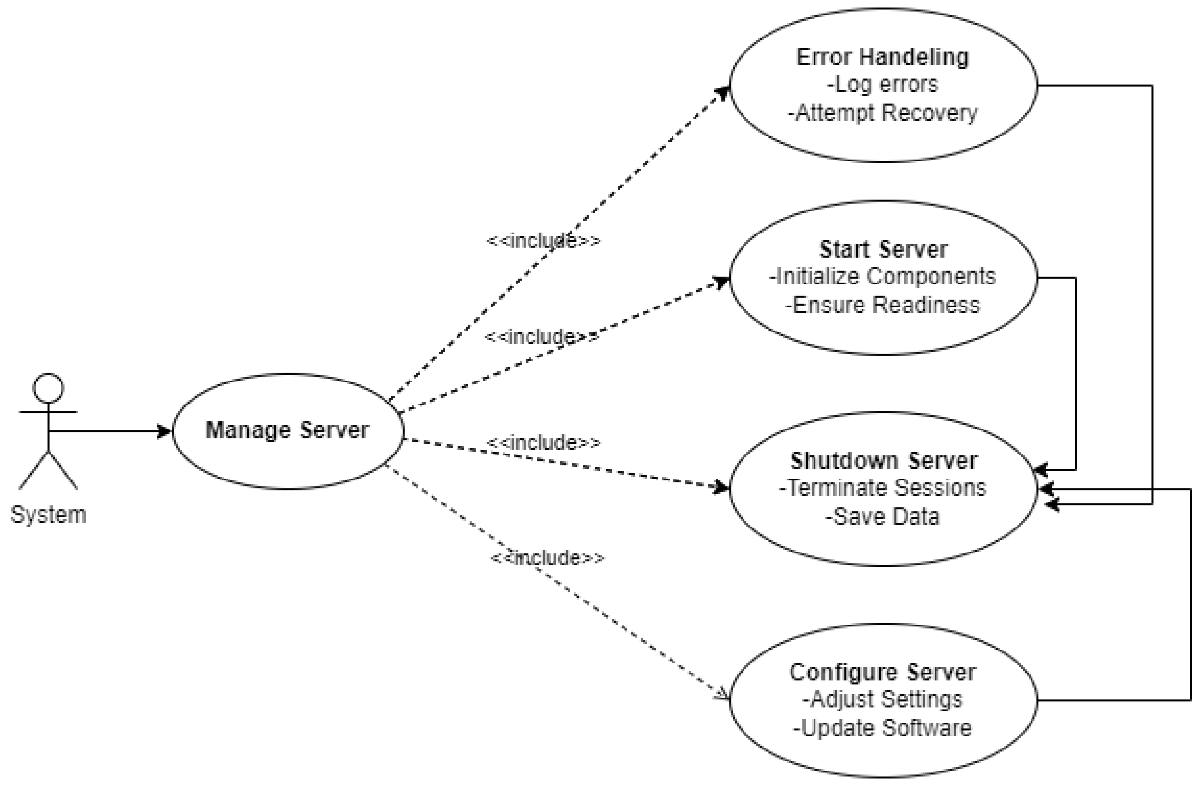
Shutdown Process:

- Gracefully terminate user sessions.
- Save unsaved data to prevent data loss.
- Release allocated resources.
- Generate shutdown logs.

Error Handling:

- Identify and log errors.
- Display user-friendly error messages.
- Automatically attempt recovery mechanisms.

UML Use Case Diagram for Boundary Conditions:



1. Manage Server

- **Description:** Represents the overall control and coordination of server-related operations.
- **Actor:** System.
- **Purpose:** Ensures smooth operation, configuration, and control of the server throughout its lifecycle.

2. Error Handling

- **Description:** Handles unexpected errors or failures during system operation.
- **Key Actions:**
    - Log errors for review and debugging.
    - Attempt recovery from errors to maintain system stability.
- **Purpose:** Minimizes downtime and ensures the system remains functional under failure conditions.

3. Start Server

- **Description:** Governs the server startup process to prepare it for operation.

- **Key Actions:**
    - Initialize critical components.
    - Ensure all systems are ready and operational.
- **Purpose:** Guarantees a successful start-up phase with all dependencies in place.

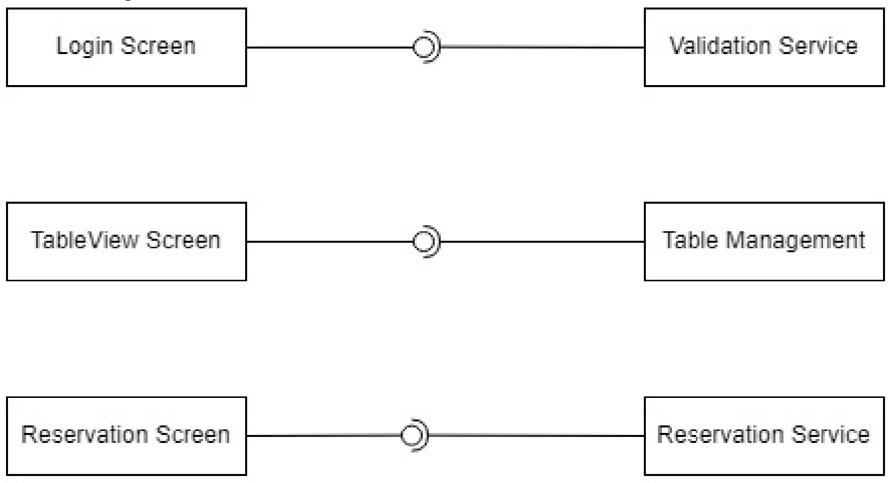
4. Shutdown Server

- **Description:** Manages the server's controlled shutdown process.
- **Key Actions:**
    - Safely terminate active sessions.
    - Save essential data to prevent loss.
- **Purpose:** Prevents data corruption and ensures graceful shutdown.

5. Configure Server

- **Description:** Allows system to customize and maintain server configurations.
- **Key Actions:**
    - Adjust system settings based on operational requirements.
    - Update software components for improved performance and security.
- **Purpose:** Ensures the server remains optimized and aligned with system requirements.

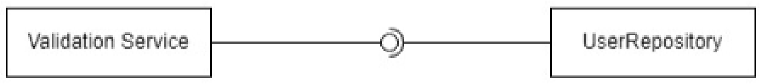
4 Subsystem Services



Login Screen: Designed to allow users to log in to the system. Collects user credentials and initiates the verification process.

TableView Screen: Allows users to view the current status of empty and occupied tables in the library and select the appropriate table.

Reservation Screen: Provides an interface for creating and updating table reservations. When the reservation is completed, it is confirmed in the system.



Validation Service: Ensures consistency of reservation and system status. For example, the LibraryManagementSystem class makes hourly updates to check the validity of reservations and send notifications for expired reservations.

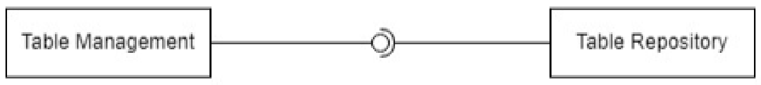
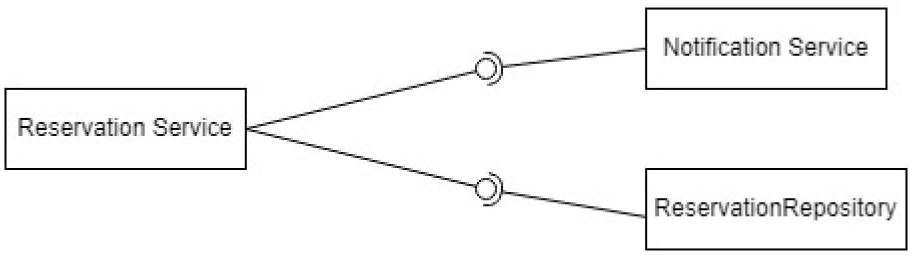


Table Service provides an interface that allows users to view and select their available tables. This provides a user-focused interaction for Desk Management.



Reservation Service: Used to create and cancel reservations. It also manages reservation information (student, table and time information). Within the system, it serves as the main component of the reservation process.

5 Glossary

1. **Library Study Desk Reservation System**
   A system designed to enable students and visitors to locate and reserve study desks within a library efficiently.

2. **Layered Architecture Pattern**
   A software design pattern that organizes a system into distinct layers (Interface, Application Logic, and Storage) to enhance modularity, maintainability, and scalability.

3. **Persistent Data Management**
   Strategies and practices for reliably storing and managing data such as user profiles, reservations, and activity logs to ensure accessibility and consistency across sessions.

4. **Entity-Relationship (ER) Diagram**
   A diagrammatic representation of data entities (e.g., User, Reservation, Table) and their relationships in a database schema.

5. **Relational Database Management System (RDBMS)**
   A type of database management system, such as MySQL or PostgreSQL, that uses structured relationships between data entities to organize and retrieve information efficiently.

6. **Data Access Layer (DAL)**
   An abstraction layer that encapsulates all interactions with the database, ensuring modularity and reducing direct dependencies on specific database technologies.

7. **SSL/TLS Protocols**
   Secure communication protocols used to encrypt data during transmission over networks, safeguarding it against unauthorized access.

8. **Scalability**
   The capability of a system to handle increased workloads or expand its capacity to meet growing demands without compromising performance.

9. **ACID Compliance**
   A set of properties (Atomicity, Consistency, Isolation, Durability) that ensure reliable and consistent database transactions.

10. **Dynamic Model**
    A model that represents the system's behavior over time, often visualized through diagrams like sequence or state diagrams.

11. **Use-Case Model**
    A framework describing how users interact with a system to achieve specific goals, presented through use-case scenarios and diagrams.

12. **Mock-Up Design**
    A preliminary visual design or prototype of a user interface, used for feedback and refinement before implementation.

13. **Functional Requirements**
    Specifications of the functionalities a system must perform, such as user authentication, desk reservation, or notifications.

14. **Nonfunctional Requirements**
    Attributes defining the quality and performance of a system, including reliability, usability, security, and scalability.

15. **Scenario**
    A narrative or detailed description of how users interact with a system to achieve a specific objective or perform a task.

16. **System Logs**
    Records of user and system activities, such as login attempts and reservation modifications, used for monitoring, debugging, and auditing.

17. **Role-Based Access Control (RBAC)**
    A security approach where access rights are assigned to users based on their roles, ensuring appropriate permissions for different user categories.

6 References

- Object-Oriented Software Engineering Using UML, Patterns,
  and JavaTM
  Third Edition

- ChatGPT

7 Appendix

- Annex – I: Distribution of Work
- Annex – II: Meeting Minutes

Distribution of Work

The responsibilities for each section of the document were distributed among the contributors as follows:

Zafer was responsible for writing Section 1.1, which defines the purpose of the system, and Section 1.2, which outlines the design goals.

Sena documented Section 2, detailing the current software architecture, including its limitations and challenges.

Aslı contributed to Sections 3.1, 3.2, and 3.3, focusing on subsystem decomposition, hardware/software mapping, and persistent data management.

Sıla developed Sections 3.4 and 3.5, addressing access control, security measures, and boundary conditions.

Ertuğrul completed Section 4, which describes the subsystem services provided by the system.

Umut reviewed the entire report, organized its structure, and finalized the document by merging all sections into a cohesive whole.

Annex – II

Meeting Minutes

Date:	28.12.2024
Location:	MS Teams
Duration:	30 Minutes
Participants:	Sıla Naz ASLAN Sena KILINÇ Aslı ÇELİK Zafer SARIK İsmail Ertuğrul BOZKURT Umut AKBAŞ

Content of the meeting (briefly explain the agenda, decisions, work distributions, etc.)

The responsibilities for each section of the document were distributed among the contributors as follows:

Zafer was responsible for writing Section 1.1, which defines the purpose of the system, and Section 1.2, which outlines the design goals.

Sena documented Section 2, detailing the current software architecture, including its limitations and challenges.

Aslı contributed to Sections 3.1, 3.2, and 3.3, focusing on subsystem decomposition, hardware/software mapping, and persistent data management.

Sıla developed Sections 3.4 and 3.5, addressing access control, security measures, and boundary conditions.

Ertuğrul completed Section 4, which describes the subsystem services provided by the system.

Umut reviewed the entire report, organized its structure, and finalized the document by merging all sections into a cohesive whole.